Cross-Validation Hand Out

Alan Arnholt

Last knit on: October 28, 2021 at 07:39:52 AM

Contents

1 Cross-Validation Handout

Note: Working definitions and graphs are taken from Ugarte, Militino, and Arnholt (2016)

1.1 The Validation Set Approach

The basic idea behind the validation set approach is to split the available data into a training set and a testing set. A regression model is developed using only the training set. Consider Figure 1 which illustrates a split of the available data into a training set and a testing set.

The percent of values that are allocated into training and testing may vary based on the size of the available data. It is not unusual to allocate 70–75% of the available data as the training set and the remaining 25–30%

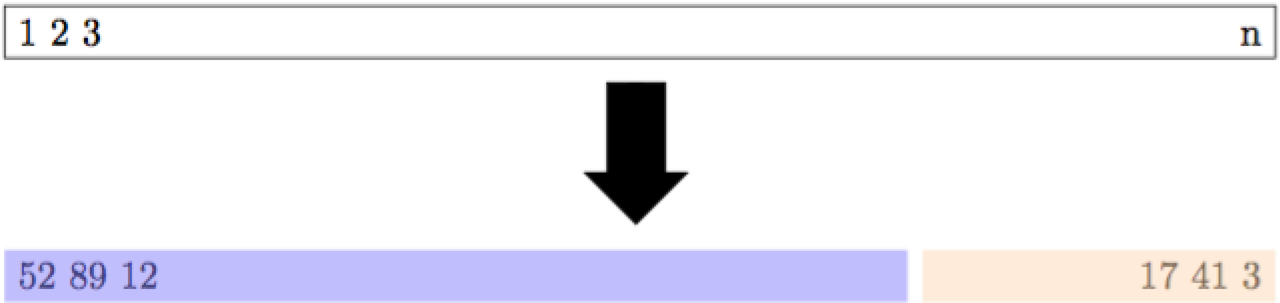


Figure 1: Validation set approach

as the testing set. The predictive performance of a regression model is assessed using the testing set. One of the more common methods to assess the predictive performance of a regression model is the mean square prediction error (MSPE). The MSPE is defined as

$$\text{MSPE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

1.2 Leave-One-Out Cross Validation

The leave-one-out cross-validation (LOOCV) eliminates the problem of variability in MSPE present in the validation set approach. The LOOCV is similar to the validation set approach as the available n observations are split into training and testing sets. The difference is that each of the available n observations are split into n training and n testing sets where each of the n training sets consist of $n-1$ observations and each of the testing sets consists of a single different value from the original n observations. Figure 2 provides a schematic display of the leave-one-out cross-validation process with testing sets (light shade) and training sets (dark shade) for a data set of n observations.

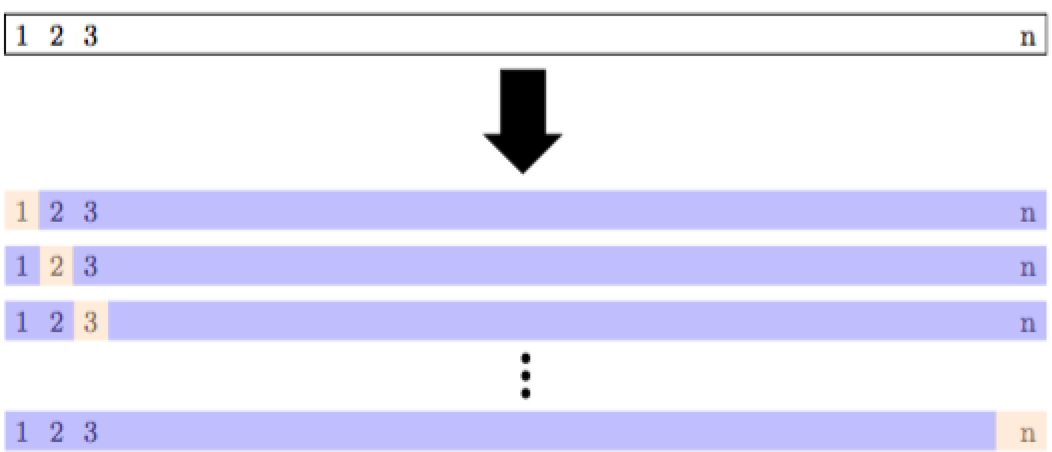


Figure 2: Leave-one-out cross validation

The MSPE is computed with each testing set resulting in n values of MSPE. The LOOCV estimate for the test MSPE is the average of these n MSPE values denoted as

$$\text{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\text{MSPE}_i \tag{2}$$

1.3 k-Fold Cross Validation

k-fold cross-validation is similar to LOOCV in that the available data is split into training sets and testing sets; however, instead of creating n different training and testing sets, k folds/groups of training and testing sets are created where $k < n$ and each fold consists of roughly n/k values in the testing set and $n - n/k$ values in the training set. Figure 3 shows a schematic display of 5-fold cross-validation. The lightly shaded rectangles are the testing sets and the darker shaded rectangles are the training sets.

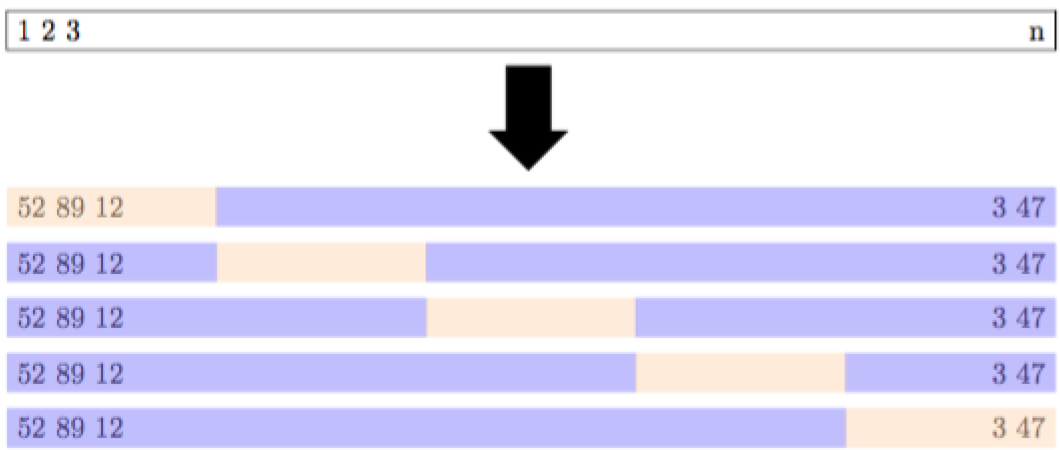


Figure 3: Five fold cross-validation

The MSPE is computed on each of the k folds using the testing set to evaluate the regression model built from the training set. The weighted average of k MSPE values is denoted as

$$\text{CV}_{(k)} = \sum_{k=1}^{k}\frac{n_k}{n}\text{MSPE}_k \tag{3}$$

Note that LOOCV is a special case of k-fold cross-validation where k is set equal to n. An important advantage k-fold cross-validation has over LOOCV is that CV_k for $k = 5$ or $k = 10$ provides a more accurate estimate of the test error rate than does CV_n.

1.4 Creating some data

```r
set.seed(357)
n <- 1000             # Number of observations to generate
SD <- 0.5
xs <- sort(runif(n, 5, 9))
ys <- sin(xs) + rnorm(n, 0, SD)
DF <- data.frame(x = xs, y = ys)
rm(xs, ys)
```

```
library(DT)
datatable(DF)
```

Show 10 entries Search:

	x	y
1	5.00218567345291	-0.461998687888378
2	5.00949552096426	-1.2552623009749
3	5.01820665318519	-0.350569477015049
4	5.02762991003692	-1.06623235222228
5	5.03651614766568	-0.0221593456286351
6	5.03874948713928	-1.7631689843736
7	5.04254586063325	-0.769226533846735
8	5.04476176574826	-1.55118888999942
9	5.04479634109885	-1.19827408822592
10	5.05191567540169	-0.278206209088567

Showing 1 to 10 of 1,000 entries Previous 1 2 3 4 5 … 100 Next

1.5 Validation Set Approach

- Create a training set using 75% of the observations in DF.
- Sort the observations in the training and testing sets.

```
n <- nrow(DF)
train <- sample(n, floor(0.75 * n), replace = FALSE)
train <- sort(train)
trainSET <- DF[train, ]
testSET <- DF[-train, ]
dim(trainSET)
```

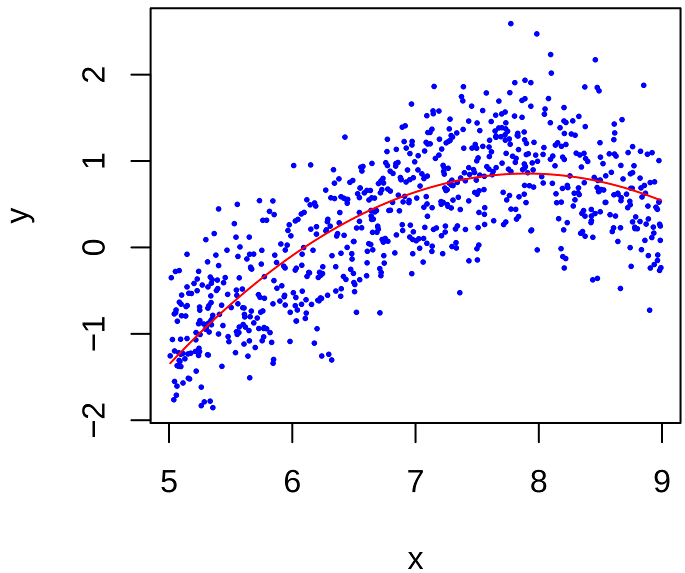
```
[1] 750   2
```

```
dim(testSET)
```

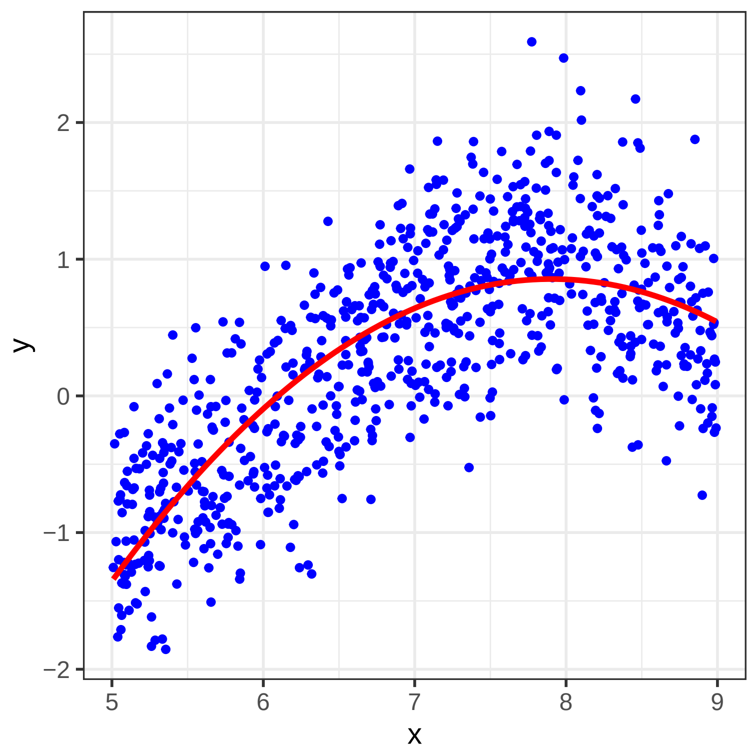
```
[1] 250   2
```

- Fit a quadratic model using the training set (trainSET).

```
library(ggplot2)
# Base R
plot(y ~ x, data = trainSET, pch = 19, cex = .25, col = "blue")
modq <- lm(y ~ poly(x, 2, raw = TRUE), data = trainSET)
yhat <- predict(modq, data = trainSET)
lines(trainSET$x, yhat, col = "red")
```




```
# ggplot2 approach
ggplot(data = trainSET, aes(x = x, y = y)) +
  geom_point(color = "blue", size = 1) +
  theme_bw() +
  geom_smooth(method = "lm", formula = y ~ poly(x, 2, raw = TRUE), color = "red", se = FALSE)
```




```
# Summary of quadratic model
summary(modq)
```

```
Call:
lm(formula = y ~ poly(x, 2, raw = TRUE), data = trainSET)
```

```
Residuals:
     Min       1Q   Median       3Q      Max 
-1.50039 -0.38306  0.00614  0.36587  1.73982 

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)    
(Intercept)             -15.52604    0.73537  -21.11   <2e-16 ***
poly(x, 2, raw = TRUE)1   4.14560    0.21471   19.31   <2e-16 ***
poly(x, 2, raw = TRUE)2  -0.26228    0.01535  -17.08   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5168 on 747 degrees of freedom
Multiple R-squared:  0.6104,    Adjusted R-squared:  0.6094 
F-statistic: 585.2 on 2 and 747 DF,  p-value: < 2.2e-16
```

1.6 Compute the training MSPE

```
MSPE <- mean(resid(modq)^2)
MSPE
```

```
[1] 0.2659911
```

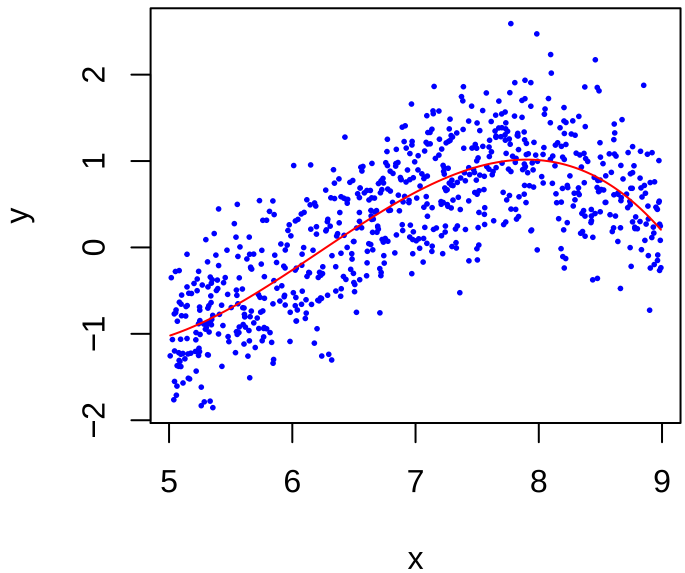
1.7 Compute the testing MSPE

```
yhtest <- predict(modq, newdata = testSET)
MSPEtest <- mean((testSET$y - yhtest)^2)
MSPEtest
```

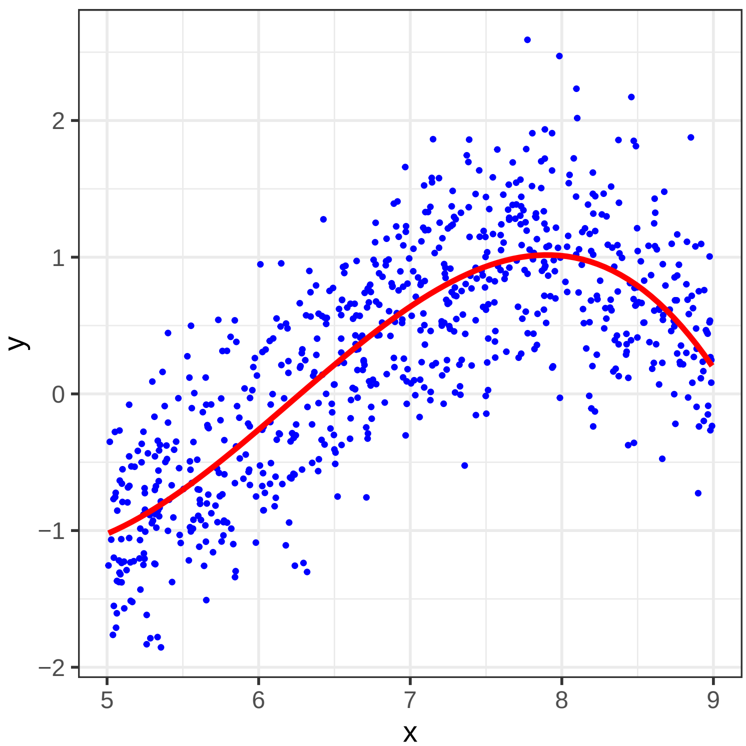
```
[1] 0.2632364
```

1.8 Fit a cubic model.

```
# Base R
plot(y ~ x, data = trainSET, pch = 19, cex = .25, col = "blue")
modc <- lm(y ~ poly(x, 3, raw = TRUE), data = trainSET)
yhat <- predict(modc, data = trainSET)
lines(trainSET$x, yhat, col = "red")
```




```r
# ggplot2 approach
ggplot(data = trainSET, aes(x = x, y = y)) +
  geom_point(color = "blue", size = 0.5) +
  theme_bw() +
  geom_smooth(method = "lm", formula = y ~ poly(x, 3, raw = TRUE), color = "red", se = FALSE)
```




```r
# Summary of cubic model
summary(modc)
```

```
Call:
lm(formula = y ~ poly(x, 3, raw = TRUE), data = trainSET)
```

```
Residuals:
     Min       1Q   Median       3Q      Max 
-1.39188 -0.34586 -0.02019  0.35683  1.58341 

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)    
(Intercept)              20.51700    4.93129   4.161 3.54e-05 ***
poly(x, 3, raw = TRUE)1 -11.85989    2.17688  -5.448 6.92e-08 ***
poly(x, 3, raw = TRUE)2   2.06478    0.31541   6.546 1.10e-10 ***
poly(x, 3, raw = TRUE)3  -0.11089    0.01501  -7.386 4.05e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4992 on 746 degrees of freedom
Multiple R-squared:  0.6369,    Adjusted R-squared:  0.6355 
F-statistic: 436.3 on 3 and 746 DF,  p-value: < 2.2e-16
```

1.9 Compute the training `MSPE`

```r
MSPE <- mean(resid(modc)^2)
MSPE
```

```
[1] 0.2478649
```

1.10 Compute the testing `MSPE`

```r
yhtest <- predict(modc, newdata = testSET)
MSPEtest <- mean((testSET$y - yhtest)^2)
MSPEtest
```

```
[1] 0.2507263
```

1.11 Your Turn

- Create a training set (80%) and testing set (20%) of the observations from the data frame `HSWRESTLER` from the `PASWR2` package. Store the results from regressing `hwfat` onto `abs` and `triceps` in the object `modf`.
- Compute the test `MSPE`.
- Note how the answers of your classmates are all different. The validation estimate of the test MSPE can be highly variable.

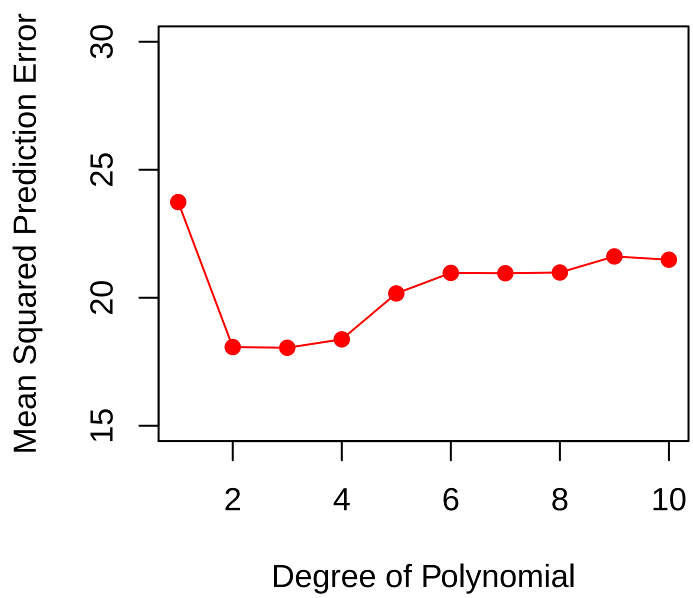
```r
# Your code here
library(PASWR2)
#
#
#
#
#
#
#
#
```

```
#
#
#
```

1.12 Your Turn

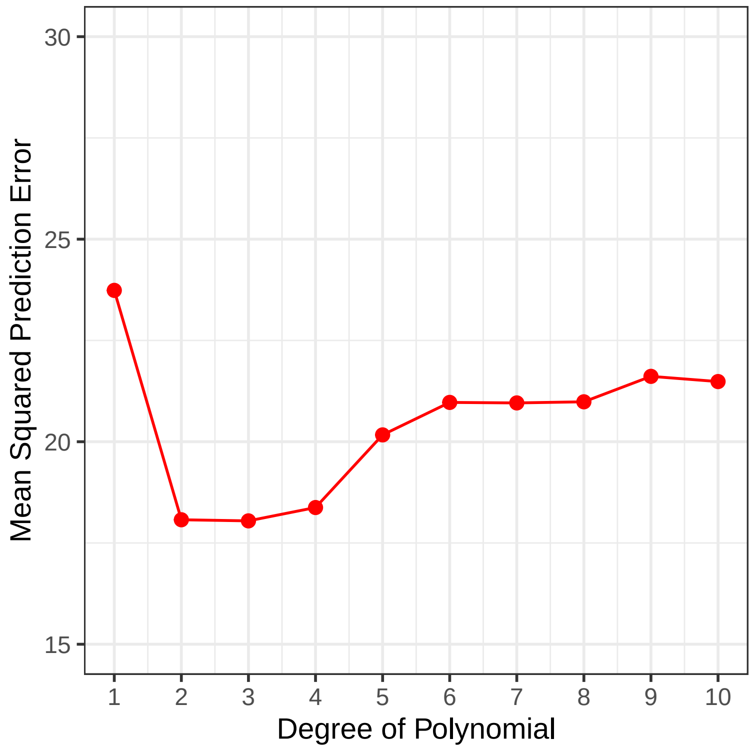
The left side of Figure 5.2 on page 178 of James et al. (2013) shows the validation approach used on the `Auto` data set in order to estimate the test error that results from predicting `mpg` using polynomial functions of `horsepower` for one particular split of the original data. The code below creates a similar graph.

```
library(ISLR)
n <- nrow(Auto)
plot(1:10, type ="n", xlab = "Degree of Polynomial", ylim = c(15, 30),
     ylab = "Mean Squared Prediction Error")
IND <- sample(1:n, size = floor(n/2), replace = FALSE)
train <- Auto[IND, ]
test <- Auto[-IND, ]
MSPE <- numeric(10)
for(i in 1:10){
    mod <- lm(mpg ~ poly(horsepower, i), data = train)
    pred <- predict(mod, newdata = test)
    MSPE[i] <- mean((test$mpg - pred)^2)
  }
lines(1:10, MSPE, col = "red")
points(1:10, MSPE, col = "red", pch = 19)
```




```
# ggplot2 approach
DF2 <- data.frame(x = 1:10, MSPE = MSPE)
ggplot(data = DF2, aes(x = x, y = MSPE)) +
  geom_point(color = "red", size = 2) +
  geom_line(color = "red") +
  theme_bw() +
  ylim(15, 30) +
  scale_x_continuous(breaks = 1:10) +
```

```
  labs(x = "Degree of Polynomial", y = "Mean Squared Prediction Error")
```



- Modify the code above to recreate a graph similar to the right side of Figure 5.2 on page 178 of James et al. (2013). Hint: Place a `for` loop before `IND`.

```
# Your code here
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
# ggplot2 approach
#
#
#
#
#
#
#
```

```
#
#
#
#
#
#
#
#
#
#
#
#
#
```

1.13 k Fold Cross Validation

- Create $k = 5$ folds.
- Compute the $\text{CV}_{k=5}$ for `modq`.

```
set.seed(1)
k <- 5
MSPE <- numeric(k)
folds <- sample(x = 1:k, size = nrow(DF), replace = TRUE)
xtabs(~folds)
```

```
folds
  1   2   3   4   5 
210 194 183 204 209 
```

```
# or
table(folds)
```

```
folds
  1   2   3   4   5 
210 194 183 204 209 
```

```
sum(xtabs(~folds))
```

```
[1] 1000
```

```
for(j in 1:k){
  modq <- lm(y ~ poly(x, 2, raw = TRUE), data = DF[folds != j, ])
  pred <- predict(modq, newdata = DF[folds ==j, ])
  MSPE[j] <- mean((DF[folds == j, ]$y - pred)^2)
}
MSPE
```

```
[1] 0.2758948 0.2656636 0.2693788 0.2378372 0.2865719
```

```
weighted.mean(MSPE, table(folds)/sum(folds))
```

```
[1] 0.2671853
```

1.13.1 Using caret

```
library(caret)
model <- train(
  form = y ~ poly(x, 2, raw = TRUE),
  data = DF,
  method = "lm",
  trControl = trainControl(
    method = "cv", number = 5
  )
)
model
```

```
Linear Regression

1000 samples
   1 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 800, 800, 800, 800, 800
Resampling results:

  RMSE       Rsquared   MAE
  0.5148039  0.6093387  0.4175265

Tuning parameter 'intercept' was held constant at a value of TRUE
```

1.14 Your Turn

- Compute the CV_8 for `modf`. Recall that `modf` was created from regressing `hwfat` onto `abs` and `triceps`.

```
# Your code here
set.seed(13)
k <- 8
MSPE <- numeric(k)
folds <- sample(x = 1:k, size = nrow(HSWRESTLER), replace = TRUE)
#
#
#
#
#
#
#
#
#
```

1.14.1 Using caret

```
model <- train(
  form = hwfat ~ abs + triceps,
  data = HSWRESTLER,
```

```
  method = "lm",
  trControl = trainControl(
    method = "cv", number = 8
  )
)
model
```

```
Linear Regression

78 samples
 2 predictor

No pre-processing
Resampling: Cross-Validated (8 fold)
Summary of sample sizes: 68, 69, 68, 68, 68, 68, ...
Resampling results:

  RMSE      Rsquared   MAE
  3.141969  0.8494259  2.510904

Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
model$results$RMSE^2
```

```
[1] 9.871968
```

1.15 Using `cv.glm` from `boot`

```
set.seed(1)
library(boot)
glm.fit <- glm(y ~ poly(x, 2, raw = TRUE), data = DF)
cv.err <- cv.glm(data = DF, glmfit = glm.fit, K = 5)$delta[1]
cv.err
```

```
[1] 0.2681278
```

1.16 Your Turn

- Compute CV_8 for `modf` using `cv.glm`. Recall that `modf` was created from regressing `hwfat` onto `abs` and `triceps`.

```
# Your code here
glm.fit <- glm(hwfat ~ abs + triceps, data = HSWRESTLER)
#
#
```

- Use `caret` and compare answers.

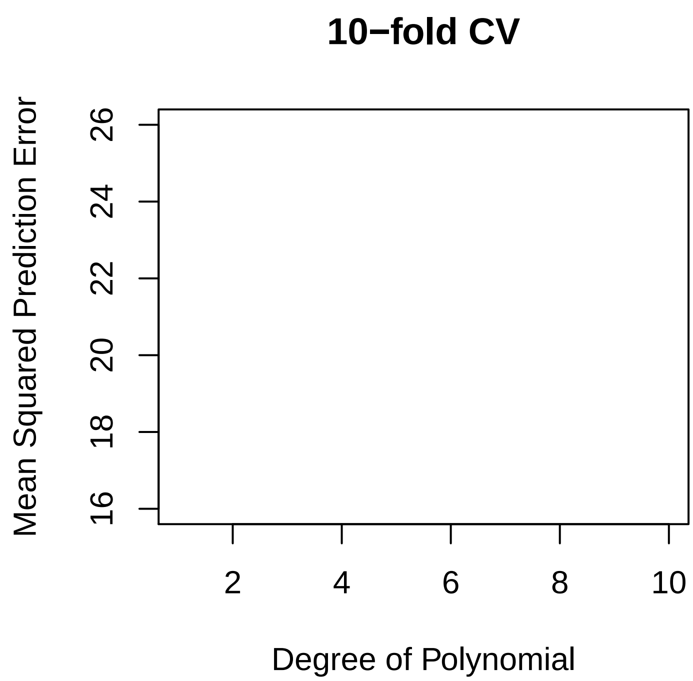
```
# Your Code Here
#
#
#
#
#
```

```
#
#
#
#
#
```

1.17 Your Turn

The right side of Figure 5.4 on page 180 of James et al. (2013) shows the 10-fold cross-validation approach used on the `Auto` data set in order to estimate the test error that results from predicting `mpg` using polynomial functions of `horsepower` run nine separate times. The code below creates a graph showing one particular run.

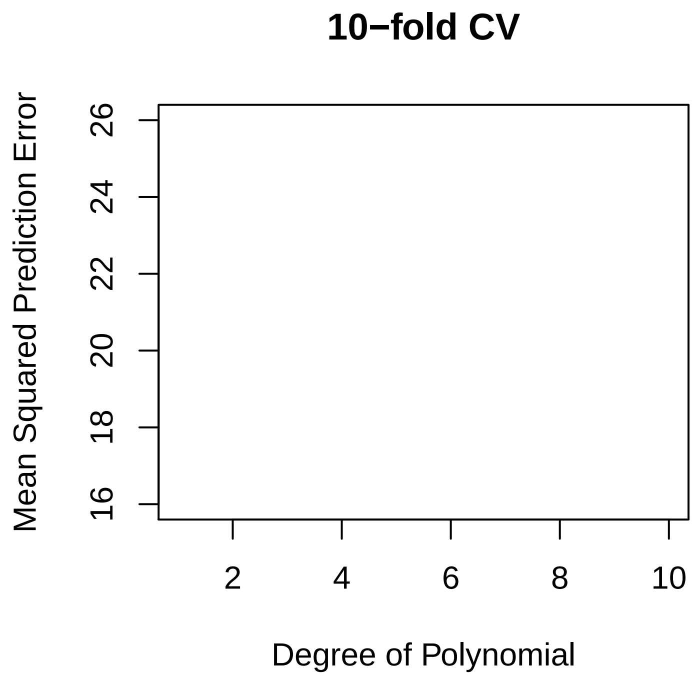
```
# Your code here
plot(1:10, type ="n", xlab = "Degree of Polynomial", ylim = c(16, 26),
     ylab = "Mean Squared Prediction Error", main = "10-fold CV")
```




```
k <- 10 # number of folds
MSPE <- numeric(k)
cv <- numeric(k)
#
#
#
#
#
#
#
#
#
#
#
```

- Use a `for` loop to run the above code nine times. The result should look similar to the right side of Figure 5.4 on page 180 of James et al. (2013).

```
# Your Code Here
set.seed(123)
plot(1:10, type ="n", xlab = "Degree of Polynomial", ylim = c(16, 26),
     ylab = "Mean Squared Prediction Error", main = "10-fold CV")
```

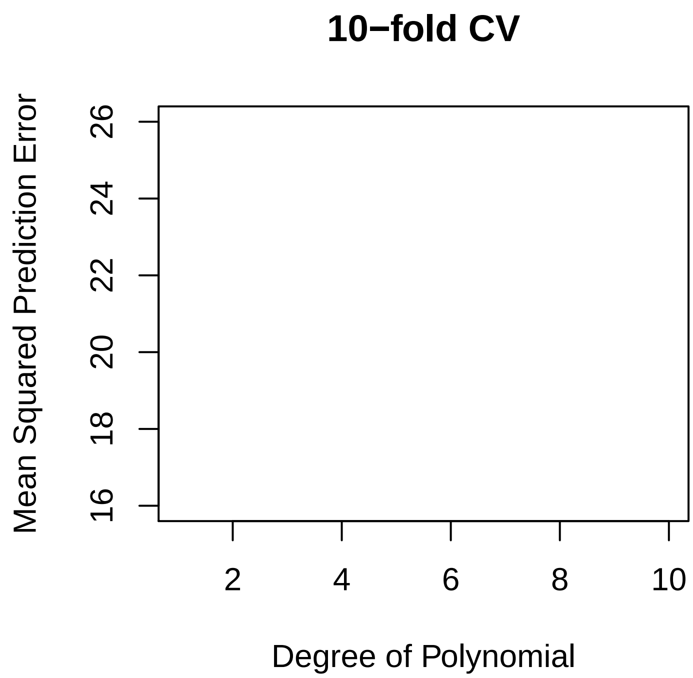



```
k <- 10       # number of folds
MSPE <- numeric(k)
cv <- numeric(k)
#
#
#
#
#
#
#
#
#
#
#
#
#
## GGplot2 approach
set.seed(123)
cv <- matrix(NA, 10, 9)
k <- 10       # number of folds
MSPE <- numeric(k)
#
#
#
#
```

```
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
```

- Use the function `cv.glm` to create similar a graph to the right side of Figure 5.4 on page 180 of James et al. (2013).

```
# Your Code Here
plot(1:10, type ="n", xlab = "Degree of Polynomial", ylim = c(16, 26),
     ylab = "Mean Squared Prediction Error", main = "10-fold CV")
```




```
#
#
#
#
#
#
#
#
#
```

```r
# GGplot2 approach
cv.err <- matrix(NA, 10, 9)
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
```

1.18 Leave-One-Out Cross-Validation

```r
set.seed(1)
k <- nrow(DF)
MSPE <- numeric(k)
folds <- sample(x = 1:k, size = nrow(DF), replace = FALSE)
# Note that replace changes to FALSE for LOOCV...can you explain why?
for(j in 1:k){
  modq <- lm(y ~ poly(x, 2, raw = TRUE), data = DF[folds != j, ])
  pred <- predict(modq, newdata = DF[folds ==j, ])
  MSPE[j] <- mean((DF[folds == j, ]$y - pred)^2)
}
mean(MSPE)
```

```
[1] 0.2663587
```

1.19 Your Turn

- Compute CV_n for `modf`. Recall that `modf` was created from regressing `hwfat` onto `abs` and `triceps`.

```r
# Your Code Here
set.seed(1)
k <- nrow(HSWRESTLER)
MSPE <- numeric(k)
#
#
#
#
#
#
#
```

Recall

$$CV_n = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_i}\right)^2$$

```
modq <- lm(y ~ poly(x, 2, raw = TRUE), data = DF)
h <- hatvalues(modq)
CVn <- mean(((DF$y - predict(modq))/(1 - h))^2)
CVn
```

```
[1] 0.2663587
```

1.20 Your Turn

- Compute CV_n for `modf` using the mathematical shortcut. Recall that `modf` was created from regressing `hwfat` onto `abs` and `triceps`.

```
# Your Code Here
modf <- lm(hwfat ~ abs + triceps, data = HSWRESTLER)
#
#
#
```

1.21 Using `cv.glm` from `boot`

- Note: If one does not use the `K` argument for the number of folds, `gv.glm` will compute LOOCV.

```
library(boot)
glm.fit <- glm(y ~ poly(x, 2, raw = TRUE), data = DF)
cv.err <- cv.glm(data = DF, glmfit = glm.fit)$delta[1]
cv.err
```

```
[1] 0.2663587
```

1.22 Your Turn

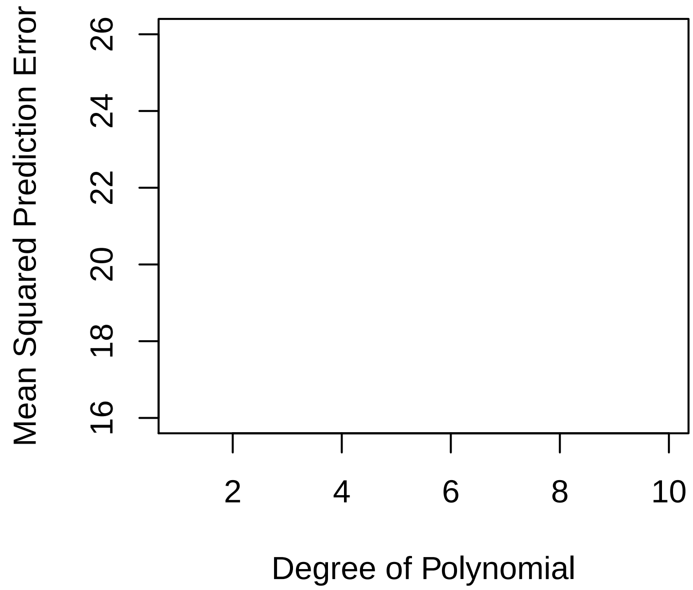
- Compute CV_n for `modf` using `cv.glm`. Recall that `modf` was created from regressing `hwfat` onto `abs` and `triceps`.

```
# Your Code Here
glm.fit <- glm(hwfat ~ abs + triceps, data = HSWRESTLER)
#
#
```

1.23 Your Turn

- Create a graph similar to the left side of Figure 5.4 on page 180 of James et al. (2013).

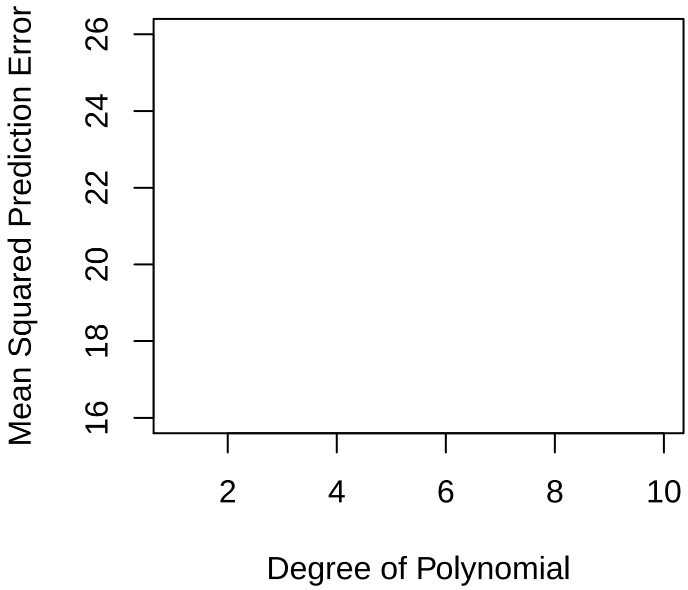
```
# Your Code Here
plot(1:10, type ="n", xlab = "Degree of Polynomial", ylim = c(16, 26),
     ylab = "Mean Squared Prediction Error")
```




```r
k <- nrow(Auto) # number of folds
MSPE <- numeric(k)
cv <- numeric(10)
#
#
#
#
#
#
#
#
#
#
#
#
```

Using the short cut formula:

```r
# Your Code Here
plot(1:10, type ="n", xlab = "Degree of Polynomial", ylim = c(16, 26),
     ylab = "Mean Squared Prediction Error")
```




```
cv <- numeric(10)
#
#
#
#
#
#
#
```

References

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani, eds. 2013. *An Introduction to Statistical Learning: With Applications in R*. Springer Texts in Statistics 103. New York: Springer.

Ugarte, María Dolores, Ana F. Militino, and Alan T. Arnholt. 2016. *Probability and Statistics with R*. Second edition. Boca Raton: CRC Press, Taylor & Francis Group.